UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/986,231	10/22/2001	William Bush	004-2928-2	4991

22120 7590 12/21/2004

ZAGORIN O'BRIEN & GRAHAM, L.L.P.
7600B N. CAPITAL OF TEXAS HWY.
SUITE 350
AUSTIN, TX 78731

EXAMINER	
STEELMAN, MARY J	

ART UNIT	PAPER NUMBER
2122	

DATE MAILED: 12/21/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/986,231	BUSH ET AL.	
	Examiner	Art Unit	
	Mary J. Steelman	2122	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *10/22/2001,12/09/2002*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-49* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-49* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *22 December 2001* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *12/09/02*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-49 are pending.

Information Disclosure Statement

2. IDS received 12/09/2002 has been considered.

Drawings

3. The drawings are objected to as failing to comply with 37 CFR 1.84(p)(5) because they include the following reference character(s) not mentioned in the description:

FIG. 1, #135, See Specification p. 6, line 20.

FIG. 3, #313, #314, #317, #318, #341, #342, #343.

FIG. 6, #690.

4. Corrected drawing sheets in compliance with 37 CFR 1.121(d), or amendment to the specification to add the reference character(s) in the description in compliance with 37 CFR 1.121(b) are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. The replacement sheet(s) should be labeled **"Replacement Sheet"** in the page header (as per 37 CFR 1.84(c)) so as not to obstruct any portion of the drawing figures. If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

Claim Objections

5. Claim 48 is objected to because of the following informalities: The claim does not end with a period, '.'. Appropriate correction is required.

5. Claims are inconsistent regarding the phrase "unused instruction positions" as disclosed in the Specification page 24, line 21.

Claim 3 recites, "unused position of the VLIW instruction"

Claim 16 recites, "unused instruction position"

Claim 21 recites "unused position in the instruction sequence"

Claim 25 recites "unused operation position of an instruction instance"

Claim 32 recites "unused component operation position of an operation instance"

Claim 35 recites "unused operation positions in each of the plural threads"

Claim 42 recites "instructions in otherwise unused operation slots"

Because the claims are referencing superscalar instructions in a long instruction word, it becomes unclear when the claim recites an instruction within the long instruction word. Furthermore, it becomes unclear when "an instruction position", "a position of an instruction instance", "an operation position", and "operation slots" are used interchangeably. It is suggested that the claims be reworded more clearly.

Double Patenting

6. Claims 1-49 are rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claims 1-36 of U.S. Patent No.6308319 B1. Although the

conflicting claims are not identical, they are not patentably distinct from each other because claims refer to similar limitations. As an example:

US Patent 6308319:

Claim 1: 'mutator code including an instance of a delay slot instruction coinciding with said safe point, wherein said delay slot instruction instance references storage encodable with an exception triggering value to trigger an exception for suspending said mutator at said safe point; executing said delay slot instruction instance...'

Application 09 / 986231:

Claim 1: 'mutator code executable by a processor and including an instruction that coincides with a safe point in an execution path...wherein based on a settable state of the processor, one of the operations of the instruction selectively triggers suspension of the mutator code at the safe point.'

Both claims reference an 'instruction with a safe point', an instruction slot with a references to a settable state (storage encodable with an exception triggering value), and triggering an exception (suspending mutator code) at a safe point dependent on the settable state value.

Similarly all independent claims reference an encoded settable state, with an exception triggering value, that is referenced by an interrupt (suspension / conditional trap) operation (instruction) which seeks to suspend mutator code at a safe point.

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 1-29, 32, and 34-49 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent 5,557,771 to Kawaguchi et al., in view of US Patent 6,738,893 B1 to Rozas.

1.

A computer program product encoded in at least one computer readable medium, the computer program product comprising:

(See FIG. 2. Col. 1, lines 16-18, "This invention relates to a data processing system having a memory unit with auxiliary memory bits (computer program product encoded), and also the protection and management of data used therein.)

-mutator code executable by a processor and including an instruction that coincides with a safe point in an execution path thereof,

(Mutator code is executing code that changes / mutates memory state (Specification page 2, lines 17-18). Any executing process executes mutator code. A safe point is a location in code where processing threads may safely allow context switching (process priority changes) without loosing program consistency- such as at the end of a basic block of instructions. This is well known.)

-the instruction explicitly encoding parallelism amongst multiple component operations thereof, (Superscalar architectures, VLIW, horizontally encoded instructions are all well known examples of 'explicitly encoded parallelism'. Rozas provided a suggestion of using VLIW encodings when scheduling to optimize memory management.)

-wherein, based on a settable state of the processor, one of the operations of the instruction selectively triggers suspension of the mutator code at the safe point.
(This limitation calls for a 'settable bit' to trigger suspension of executing threads at a safe point. See Kawaguchi, (col. 1, line 65-col. 2, line 5), "...when the data processor is going to read or write the value of a data word, the processing of the data processor is interrupted according to the value of the corresponding auxiliary memory bit (settable state of the processor), and a predetermined process is started...a decision (decide whether it is a safe point to interrupt) is made whether to interrupt (trigger suspension of mutator code) or continue the read or update operation...")

Official Notice is taken 'encoding parallelism', 'mutator code', and compiler knowledge of 'safe points' to permit process interrupts are well known in the art. Mutator code is merely code that is changing memory, virtually every executing program. Safe points are merely locations in code where threads are consistent, safe for context switching, such as at the end of a basic block. Any well behaved program would manage thread activity and interrupts. Therefore, it would have been obvious, to modify Kawaguchi's invention regarding memory accesses, to include the

these features, and to include encoded parallelism as disclosed by Rozas, when referencing a value / memory bit to determine whether to interrupt and proceed with a predetermined process.

2.

-the suspension triggering operation includes a conditional trap operation;

(Official notice is taken that a conditional trap is a type of interrupt. Kawaguchi disclosed interrupts (col. 3, lines 1-3), "...checks the values of the auxiliary memory bits, and according to the set value, sends a processing interruption signal...")

-wherein the settable state is encoded in storage accessed on execution by the conditional trap operation.

(Kawaguchi disclosed the settable state is encoded. Col. 8, lines 15-16, "FIG. 11 shows the setting of values for auxiliary memory bits (state is encoded)..." Col. 3, lines 1-3, "...checks the values of the auxiliary memory bits (storage accessed on execution), and according to the set value, sends a processing interruption (trap operation) signal..."

3.

-wherein the processor includes a very long instruction word-type (VLIW) processor and wherein the instruction is a VLIW instruction executable thereby;

(Official Notice is taken that a VLIW is a type of word instruction, as disclosed by Kawaguchi, whereby multiple instructions are packed into one long word. Multiple slots hold individual

instructions, which are issued in sequence, as long as there is no data dependence. This is well known in the art.

Kawaguchi failed to disclose multiple processors which handle VLIW. However Rozas disclosed VLIW (col. 1, line 11) instruction scheduling. Col. 4, lines 54-63, "...commands...available for inclusion in an instruction word are reviewed by the scheduling process...constraints...are satisfied...the scheduler selects a command which does not depend on the results of any other command...")

-wherein the suspension triggering operation includes a conditional trap operation encoded in an otherwise unused position of the VLIW instruction.

(Kawaguchi disclosed (col. 1, line 66-col. 2, line 2), "processor is interrupted (conditional trap operation / suspension triggering operation) according to the value of the corresponding auxiliary memory bit, and a predetermined process is started." Official Notice is taken that compilers attempt to pack instructions in unused positions of instruction words, as long as the arrangement does not interfere with data dependence situations. This is well known in the art.

Kawaguchi failed to disclose VLIWs. However Rozas disclosed (Abstract, lines 1-2), "scheduling...execution of operations in a plurality of instruction word formats..." and VLIW scheduling (col. 3, line 43).

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention to have modified Kawaguchi's invention to include specific details related to scheduling VLIW processors, as they are well known in the art. Kawaguchi's invention to

manage the memory unit with an auxiliary memory bits (col. 1, lines 16-18) may equally well be used with a multiprocessor system.

4.

-wherein the mutator code is executable by the processor as plural execution threads each with respective safe points defined therein;

(Kawaguchi failed to disclose a plurality of threads. Official Notice is taken that execution with a plurality of threads is well known in the art. Threads must have defined 'safe points' to allow proper execution.)

-wherein the settable state is encoded in one or more storage locations accessible to the execution threads;

(Kawaguchi disclosed a settable state. Col. 2, line 67-col. 3, line 3, "The control unit also checks the values (settable state) of the auxiliary memory bits (accessible to the execution threads) , and according to the set value, sends a processing interruption signal to the system bus.")

-wherein each of the execution threads includes an instance of the suspension triggering operation coinciding with respective of the safe points.

(Col. 5, lines 2-34, "...a sequence of steps to take place that when a procedure belonging to the operating system is in execution...the privilege state...is notified to the memory unit with auxiliary memory bits through a signal line...and the current operation is completed...according to the state of a register in the control unit...")

5.

-the mutator code is multi-threaded.

(Rozas disclosed multi-threaded code, col. 1, line 11, 'VLIW'.)

6.

-suspension code executable by the processor and responsive to the triggering operation, wherein upon execution, the suspension code advances the mutator code to the safe point.

(Kawaguchi disclosed, col. 5, line 33, "operation is completed according to the state of a register in the control unit." Kawaguchi disclosed permissions. Priority level of threads is well known in the art. Threads complete to a safe point before priority is given to another thread.)

7.

-the suspension triggering operation is encoded at the safe point.

Kawaguchi failed to disclose specific details related to the scheduling of operations. However (Rozas disclosed scheduling (col. 4, lines 54-63) that considers necessary constraints. "All of the commands which are available for inclusion in an instruction word are reviewed by the scheduling process...the scheduler selects a command which does not depend on the results of any other command (not data dependent, a safe point in the code).")

Therefore it would have been obvious, to one of ordinary skill in the art, at the time of the invention to include details related to scheduling a triggering operation (an exception) at a safe point in the execution of the code, because that is the only way to ensure proper code execution.

8.

-the suspension triggering operation is encoded at a position in an execution sequence of the mutator code, wherein the position precedes the safe point; and wherein the mutator code is advanced to the safe point.

(Rozas disclosed scheduling (col. 4, lines 54-63) that considers necessary constraints. If a suspension is encountered prior to a safe point, code must continue to a safe point to provide proper execution.)

Therefore it would have been obvious, to one of ordinary skill in the art, at the time of the invention to include details related to scheduling a triggering operation (an exception) at a safe point in the execution of the code, because that is the only way to ensure proper code execution.

9.

-suspension code executable by the processor and responsive to the triggering operation, wherein upon execution, the suspension code supplies a collector with data that identifies zero or more storage locations containing pointers in use by the mutator code at the safe point.

(Kawaguchi disclosed, col. 2, line 67-col. 3, line 3, "The control unit also checks the values of the auxiliary memory bits (storage locations containing pointers in use by the mutator code at the safe point), and according to the set value, sends a processing interruption signal (suspension code) to the system bus.")

10.

-comprising: collector code executable by the processor to encode the settable state.

(Kawaguchi, FIG. 11, col. 8, lines 15-16, "FIG. 11 shows the setting of values for auxiliary memory bits...")

11.

-the collector code includes first and second portions, the first portion ancillary to dynamic memory allocation, operation of the first portion causing the settable state to encode an exception triggering value in response to a garbage collection desired state of dynamic memory,

-the second portion encoding garbage collection operations to be performed after suspension of the mutator code at the safe point.

(Kawaguchi, col. 1, line 65-col. 2, line 5, "....processor is interrupted according to the value of the corresponding auxiliary memory bit (a settable state), and a predetermined process is started (garbage collection)...a decision is made whether to interrupt of continue the read or update operation (until a safe point)."

Official Notice is given that while Kawaguchi did not disclose 'garbage collection', it is a well known memory management technique that is performed a various times after a system interrupt.

12.

-the settable state is encoded in a register accessible to the processor on execution of the suspension triggering operation.

(Kawaguchi, col. 2, lines 34-35, "FIG. 11 is a diagram showing the setting of auxiliary memory bits for relation of data.")

13.

-the settable state is encoded in one or more bits of a processor status register.

(Kawaguchi, col. 2, lines 34-35, "FIG. 11 is a diagram showing the setting of auxiliary memory bits for relation of data.")

14.

-the at least one computer readable medium is selected from the set of a disk, tape or other magnetic, optical, electronic or semiconductor storage medium and a network, wired, wireless or other communications medium.

(Kawaguchi, col. 2, lines 11-12, "FIG. 1 is a block diagram of an embodiment of ;the present invention.", col. 3, lines 18-21, "...memory unit...is used for storing programs and data (semiconductor storage medium)...")

15.

A method of suspending a mutator at a safe point, the method comprising:

-executing mutator code including an instance of an instruction coinciding with the safe point,

-wherein the instruction instance references storage encodable with an exception triggering value to trigger an exception for suspending the mutator at the safe point;

-executing the instruction instance without the exception triggering value encoded in the storage;

-in response to a start garbage collection event, encoding the storage with the exception triggering value,

-thereafter executing the instruction instance, thereby triggering the exception;

-in response to the exception, suspending the mutator at the safe point.

(Kawaguchi disclosed executing mutator code that references encoded storage, thereby determining whether to execute an exception and related code. Col. 1, line 65-col. 2, line 5, "...data processor is going to read or write (mutator code)...the processing of the data processor is interrupted according to the value of the corresponding auxiliary memory bit (references storage encodable with an exception triggering value), and a predetermined process is started (garbage collection)...a decision is made whether to interrupt or continue (depending on the permissions / thread priority proceed to a safe point in the code before the interrupt) the read or update operation..."

Kawaguchi failed to disclose 'garbage collection'. Official Notice is given that 'garbage collection' is a well known type of interrupt, used in memory management. Rozas disclosed more details related to efficient scheduling that considers constraints (col. 4, lines 54-63). Rozas disclosed (col. 7, lines 5-10), "eliminating a significant portion of the no-op instructions...scheduling commands into instruction words based on both execution time and space considerations.")

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have included 'garbage collection' as a likely type of exception causing suspension of mutator code.

16.

-the instruction instance coinciding with the safe point is encoded in an otherwise unused instruction position.

(Kawaguchi failed to disclose details related to scheduling instructions. A long instruction word can handle multiple instructions as long as there is no data interference. Rozas disclosed more details related to efficient scheduling that considers constraints (col. 4, lines 54-63). Rozas disclosed (col. 7, lines 5-10), "eliminating a significant portion of the no-op instructions...scheduling commands into instruction words based on both execution time and space considerations.")

Therefore, it would have been obvious to one of ordinary skill in the art, at the time of the invention, to include details as disclosed by Rozas, regarding the efficient scheduling of instructions by using an otherwise unused instruction position thereby optimizing code for size and speed.

17.

-the otherwise unused instruction position includes an otherwise unallocated instruction position within a very long instruction word type (VLIW) instruction.

(See limitations addressed in claim 3 above.)

18.

-the otherwise unused instruction position includes a delay slot of a delayed control transfer instruction.

Official Notice is given that the "delay slot of a delayed control transfer instruction" is typically a no-op instruction that allows the system a cycle to handle data in process from the control transfer instruction. Rozas provided the suggestion to eliminating a significant portion of the no-op instructions" thereby optimizing the scheduling of instructions.

Therefore, it would have been obvious to one of ordinary skill in the art, at the time of the invention, to include details as disclosed by Rozas, regarding the efficient scheduling of instructions by using an otherwise unused instruction position thereby optimizing code for size and speed.

19.

-the instruction instance coinciding with the safe point is encoded in a VLIW instruction position.

(See limitations addressed in claim 3 above.)

20.

-the instruction instance coinciding with the safe point is encoded in a delay slot of a branch instruction.

(See limitations addressed in claim 3 above. A delay slot of a branch is a merely an example of an "otherwise unused position of the VLIW instruction.")

21.

A computer program product encoded in computer readable media, the computer program product comprising:

-an instruction sequence including an operation that coincides with a safe point therein,

-wherein the operation is encoded in an otherwise unused position in the instruction sequence and is executable at least partially in parallel with one or more operations of the instruction sequence, the operation referencing a state that selectively triggers suspension of the instruction sequence at the safe point.

(Kawaguchi disclosed executing mutator code that references encoded storage, thereby determining whether to execute an exception and related code. Col. 1, line 65-col. 2, line 5, "...data processor is going to read or write (mutator code)...the processing of the data processor is interrupted according to the value of the corresponding auxiliary memory bit (references storage encodable with an exception triggering value), and a predetermined process is started (garbage collection)...a decision is made whether to interrupt or continue (depending on the permissions / thread priority proceed to a safe point in the code before the interrupt) the read or update operation..."

Official Notice is given that an "unused position in the instruction sequence" is typically a no-op instruction that allows the system a cycle to handle data. Rozas provided the suggestion to

eliminating a significant portion of the no-op instructions" thereby optimizing the scheduling of instructions.

Therefore, it would have been obvious to one of ordinary skill in the art, at the time of the invention, to include details as disclosed by Rozas, regarding the efficient scheduling of instructions by using an otherwise unused instruction position thereby optimizing code for size and speed.

22.

-the instruction sequence includes a delayed control transfer instruction; and wherein the otherwise unused position is a delay slot of the delayed control transfer instruction.
(See limitations addressed in claim 3 above.)

23.

-the instruction sequence includes explicitly parallel instruction encodings; and wherein the otherwise unused position is an operation position of one of the explicitly parallel instruction encodings.
(See rejection of claim 3 above. A VLIW is an example of 'explicitly parallel instruction encoding.")

24.

-the instruction sequence includes very long instruction word-type (VLIW) instructions; and wherein the otherwise unused position is an operation position of one of the VLIW instructions. (See rejection of claim 3 above. A VLIW is an example of 'explicitly parallel instruction encoding.")

25.

A computer-implemented method of providing mutator code with support for suspension of at least one execution thread thereof at a safe point, the method comprising:

-encoding information accessible to a collector process to identify at least a portion of a root set of storage locations at a safe point in the execution sequence of instructions;

-encoding in an otherwise unused operation position of an instruction instance that coincides with the safe point, a conditional trap operation that references storage encodable with an exception triggering value to trigger an exception for suspending execution of the mutator code at the safe point.

(Kawaguchi disclosed executing mutator code that references encoded storage, thereby determining whether to execute an exception and related code. Col. 1, line 65-col. 2, line 5, "...data processor is going to read or write (mutator code)...the processing of the data processor is interrupted according to the value of the corresponding auxiliary memory bit (references storage encodable with an exception triggering value), and a predetermined process is started (garbage collection)...a decision is made whether to interrupt or continue (depending on the

permissions / thread priority proceed to a safe point in the code before the interrupt) the read or update operation..."

Official Notice is given that an "unused position in the instruction sequence" is typically a no-op instruction that allows the system a cycle to handle data. Rozas provided the suggestion to eliminating a significant portion of the no-op instructions" thereby optimizing the scheduling of instructions.

Therefore, it would have been obvious to one of ordinary skill in the art, at the time of the invention, to include details as disclosed by Rozas, regarding the efficient scheduling of instructions by using an otherwise unused instruction position thereby optimizing code for size and speed.

26.

-the identified portion of the root set includes zero or more registers and stack locations containing pointers in use by the mutator process at the safe point.

(Kawaguchi, FIGs. 1-4)

27.

-a remaining portion of the root set includes storage maintained by an execution environment for the mutator code.

(Kawaguchi, FIGs. 1-7)

28.

-compiling object code from source code, the conditional trap operation instance being encoded in the object code as a result of the compiling.

(Kawaguchi, col. 2, lines 1-2, "...a predetermined process is started...(an exception event is compiled into the executing code)")

29.

-supplying the execution sequence of instructions via at least one computer readable medium selected from the set of a disk, tape or other magnetic, optical, electronic or semiconductor storage medium and a network, wired, wireless or other communications medium.

(See limitations addressed in claim 14 above.)

32.

A computer program product comprising:

-a compiler that generates an execution sequence of instructions including an explicit encoding of parallelism amongst component operations thereof,

-which encodes in an otherwise unused component operation position an operation instance that references storage encodable with an exception triggering value to trigger an exception when the operation instance is executed as part of a mutator

-to thereby suspend execution of the mutator at a safe point;

-a map generator that supplies a collector process with information identifying a root set of storage locations in use by the mutator for pointer storage at the safe point, the safe point coinciding with the operation instance.

(Kawaguchi disclosed executing mutator code that references encoded storage, thereby determining whether to execute an exception and related code. Col. 1, line 65-col. 2, line 5, "...data processor is going to read or write (mutator code)...the processing of the data processor is interrupted according to the value of the corresponding auxiliary memory bit (references storage encodable with an exception triggering value), and a predetermined process is started (garbage collection)...a decision is made whether to interrupt or continue (depending on the permissions / thread priority proceed to a safe point in the code before the interrupt) the read or update operation...")

Official Notice is given that an "unused position in the instruction sequence" is typically a no-op instruction that allows the system a cycle to handle data. Rozas provided the suggestion to 'encode in an otherwise unused component operation position', eliminating a significant portion of the no-op instructions (col. 6, lines 60-61) thereby optimizing the scheduling of instructions for (col. 1, line 11) a very long instruction word (VLIW) (explicit encoding of parallelism).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time of the invention, to include details as disclosed by Rozas, regarding the efficient scheduling of

instructions by using an otherwise unused instruction position thereby optimizing code for size and speed.

34.

-encoded by or transmitted in at least one computer readable medium selected from the set of a disk, tape or other magnetic, optical, semiconductor or electronic storage medium and a network, wireline, wireless or other communications medium.

(See limitations addressed in claim 14 above.)

35.

A method of advancing plural threads to coordination points in respective execution paths thereof, the method comprising:

-encoding an exception triggering value in storage referenced by respective instances of one or more operations encoded in respective otherwise unused operation positions in each of the plural threads;

-for each of the plural threads, suspending execution thereof in response to execution of a respective operation instance, the respective operation instance coinciding with one of the coordination points therein.

(Kawaguchi disclosed executing mutator code that references encoded storage (exception triggering value), thereby determining whether to execute an exception and related code. Col. 1, line 65-col. 2, line 5, "...data processor is going to read or write (mutator code)...the processing of the data processor is interrupted (suspending execution in response to execution of a operation

instance) according to the value of the corresponding auxiliary memory bit (references storage encodable with an exception triggering value), and a predetermined process is started ...a decision is made whether to interrupt or continue (depending on coordination of threads) the read or update operation...")

Official Notice is given that an "unused operation position" is typically a no-op instruction that allows the system a cycle to handle data. Rozas provided the suggestion to 'encode in an otherwise unused component operation position', eliminating a significant portion of the no-op instructions (col. 6, lines 60-61) thereby optimizing the scheduling of instructions for (col. 1, line 11) a very long instruction word (VLIW) (plurality of threads).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time of the invention, to include details as disclosed by Rozas, regarding the efficient scheduling of instructions by using an otherwise unused instruction position thereby optimizing code for size and speed.

36.

-the unused operation positions include delay slots of respective delayed control transfer instructions.

(See limitations addressed in claim 3 above.)

37.

-the unused operation positions include operation positions of respective explicitly parallel instruction encodings.

(See limitations addressed in claim 3 above.)

38.

-the unused operation positions include operation slots of respective very long instruction word-type (VLIW) instructions.

(See limitations addressed in claim 3 above.)

39.

-the coordination points include synchronization points for thread state synchronization amongst the plural threads.

(See limitations addressed in claim 4 above.)

40.

-the coordination points include safe points at which respective of the plural threads have a consistent state.

(See limitations addressed in claim 4 above.)

41.

-the coordination points include safe points at which information descriptive of those temporary storage locations containing references to dynamically-allocated memory in the context of each

function in a calling hierarchy of functions of a respective of the plural threads ascertainable by a memory reclamation component for use in defining a root set of references to the dynamically-allocated memory.

(See limitations addressed in claim 4 above.)

42.

A method of coordinating garbage collection with execution of a multithreaded mutator, wherein the garbage collection is performed at safe points in an execution trajectory of the multi-threaded mutator, and wherein potentially inconsistent threads of the multi-threaded mutator are suspended at the safe points to facilitate the garbage collection, the method comprising:

-upon a start garbage collection event, encoding an exception triggering value in storage referenced by exception triggering instructions in otherwise unused operation slots of instruction encodings that coincide with the safe points;

-thereafter, upon execution of the exception triggering instructions, suspending the corresponding one of the threads;

-performing the garbage collection after each of the threads is suspended.

(Kawaguchi disclosed executing mutator code that references encoded storage, thereby determining whether to execute an exception and related code. Col. 1, line 65-col. 2, line 5, "...data processor is going to read or write (mutator code)...the processing of the data processor is interrupted according to the value of the corresponding auxiliary memory bit (references storage encodable with an exception triggering value), and a predetermined process is started

(garbage collection)...a decision is made whether to interrupt or continue (depending on the permissions / thread priority proceed to a safe point in the code before the interrupt) the read or update operation..."

Official Notice is given that an "unused position in the instruction sequence" is typically a no-op instruction that allows the system a cycle to handle data. Kawaguchi failed to specifically disclose 'garbage collection', however 'garbage collection' is merely a specific example of an exception. Rozas provided the suggestion to 'encode in an otherwise unused component operation position', eliminating a significant portion of the no-op instructions (col. 6, lines 60-61) thereby optimizing the scheduling of instructions for (col. 1, line 11) a very long instruction word (VLIW) (explicit encoding of parallelism).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time of the invention, to include details as disclosed by Rozas, regarding the efficient scheduling of instructions by using an otherwise unused instruction position thereby optimizing code for size and speed.

43.

-the instruction encodings include one of component operations of a very long instruction word-type (VLIW) instruction; and a pipelined first operation and corresponding delay slot operation. (See limitations addressed in claim 3 above.)

44.

-each of the threads is consistent when suspended at its corresponding safe point.

(See limitations addressed in claim 4 above.)

45.

-the multi-threaded mutator is prepared such that dynamically allocated storage locations reachable by a particular of the threads are identifiable at the safe point thereof.

(See limitations addressed in claim 4 above.)

46.

-the multi-threaded mutator is compiled and includes storage maps emitted by a compiler,

-the storage maps identifying dynamically allocated storage locations reachable by the multi-threaded mutator at the safe points.

(Kawaguchi: See tables FIG. 13A-16 – data relating tables.)

47.

An apparatus comprising:

-a processor having a settable state accessible to plural execution threads thereon and having an instruction set that includes an exception triggering operation encodable in an operation position of an instruction encoding for parallel execution of component operations;

-media accessible by the processor for encoding mutator code that includes an instance of the instruction encoding with an instance of the exception triggering operation encoded therein,

-the exception triggering operation instance referencing the settable state; and a handler responsive to execution of the exception triggering operation instance when the settable state encodes the exception triggering value.

(Kawaguchi disclosed executing mutator code that references encoded storage , thereby determining whether to execute an exception and related code. Col. 1, line 65-col. 2, line 5, "...data processor is going to read or write (mutator code)...the processing of the data processor is interrupted according to the value of the corresponding auxiliary memory bit (settable state) ...a predetermined process is started (instance of the exception triggering operation encoded therein)...a decision is made whether to interrupt or continue (responsive to execution of the exception triggering operation) the read or update operation..."

Official Notice is given that an "instruction encoding for parallel execution " is typical of encoding for a VLIW as disclosed by Rozas (col. 1, line 11). Rozas provided the suggestion to encode by optimizing the scheduling of instructions for (col. 1, line 11) a very long instruction word (VLIW) (explicit encoding of parallelism).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time of the invention, to include details as disclosed by Rozas, regarding the efficient scheduling of instructions by encoding parallel execution in a VLIW, thereby optimizing code for size and speed.

48.

-thread suspension means responsive to a start garbage collection event to encode the settable state with an exception triggering value; and

(See limitations addressed in claim 11 above.)

49

-the exception triggering instruction is selected from the set of a trap on condition code instruction, a tagged arithmetic instruction, a precise interrupt generating instruction, an exception triggering instruction used in the mutator code solely for thread suspension, and an exception triggering used in the mutator code for purposes other than thread suspension but with support in the exception handler for distinguishing between uses.

9. Claims 30, 31, and 33 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent 5,557,771 to Kawaguchi et al., in view of US Patent 6,738,893 B1 to Rozas, and further in view of US PreGrant Publication 2002/0029357 A1 to Charnell et al.

30.

-receiving a source code precursor of the mutator code into an execution environment therefor, the execution environment including a Just In Time (JIT) compiler, wherein the conditional trap operation instance encoding and the information encoding are performed by the JIT compiler.

The Kawaguchi / Rozas combination failed to disclose a JIT compiler. However, Charnell disclosed memory management (Abstract, line 1), optimizing compilers [0019], and a [0022] "Just-in-Time (JIT) compiler...execution of the code, the execution is stopped..." [0026], "For the JIT compiler, when the 'invoke' instruction for a method is encountered, control is passed to the JIT compiler..." Official Notice is taken that Kawaguchi's invention could have been modified to accommodate a JIT compiler, as they are well known in the art for receiving source code and encoding an operation (including any trap / exception) as encountered.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have compiled code using a JIT compiler as it is well known in the art and effective at compiling / scheduling code, including conditional trap code.

31.

-the source code precursor receiving is via at least one computer readable medium selected from the set of a disk, tape or other magnetic, optical, electronic or semiconductor storage medium and a network, wired, wireless or other communications medium.

(See limitations addressed in claim 14 above.)

33.

-wherein the compiler includes one of a batch compiler; and a just-in-time type (JIT) compiler.

(See limitations addressed in claim 30 above.)

Conclusion

10. This Application contains broader limitations than previously issued patent 6,308,319 to Bush. However, prior art discloses optimizing instruction encoding by using no-op positions of 'explicitly parallel instructions' comprising a word. An instruction may be scheduled by a compiler at any location where a data dependency does not exist. Compilers have rules and constraints that must be enforced. It would be obvious to schedule an exception / trap / garbage collection at point in the code where the threads are consistent. It is obvious that memory management, including garbage collection is scheduled a suitable points in the code.

11. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Mary Steelman, whose telephone number is (571) 272-3704. The examiner can normally be reached Monday through Thursday, from 7:00 AM to 5:30 PM If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached at (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Mary Steelman

12/09/2004

WEI Y. ZHEN
PRIMARY EXAMINER